Zero shot learning

Deep Learning (DSE316/616)

Vinod K Kurmi
Assistant Professor, DSE

Indian Institute of Science Education and Research Bhopal

Nov 10, 2022

Disclaimer

- Much of the material and slides for this lecture were borrowed from
  - Bernhard Schölkopf's MLSS 2017 lecture,
  - Tommi Jaakkola's 6.867 class,
  - CMP784: Deep Learning Fall 2021 Erkut Erdem Hacettepe University
  - Fei-Fei Li, Andrej Karpathy and Justin Johnson's CS231n class
  - Hongsheng Li's ELEG5491 class
  - Tsz-Chiu Au slides
  - Mitesh Khapra Class notes
  - Vineeth Balasubramanian Class
  - Nikita Detkov slides

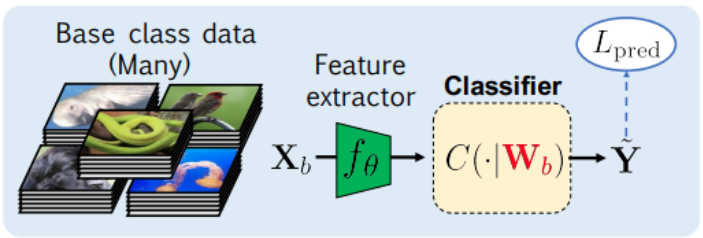
Learning with Limited Supervision

Problem

- Deep learning model ==> Heavily depends upon training ***labeled data***.
- Not directly suited to learn from few samples

Solution

- Trained model capable of rapidly generalizing to new tasks with only a few samples.
- Enable models to perform under practical scenarios.



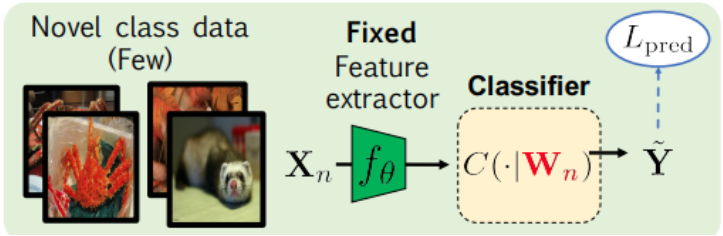


Image credit: Chen et al A Closer Look at Few-shot Classification ICLR 2019

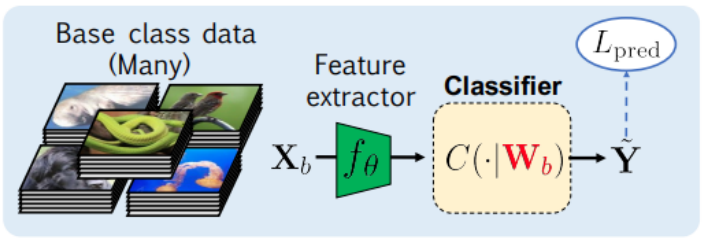
Learning with Limited Supervision

Problem

- Deep learning model ==> Heavily depends upon training ***labeled data***.
- Not directly suited to learn from few samples

Solution

- Trained model capable of rapidly generalizing to new tasks with only a few samples.
- Enable models to perform under practical scenarios.



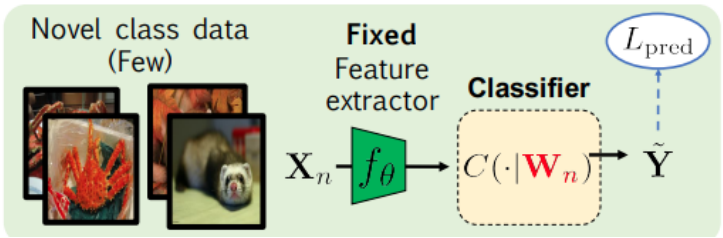
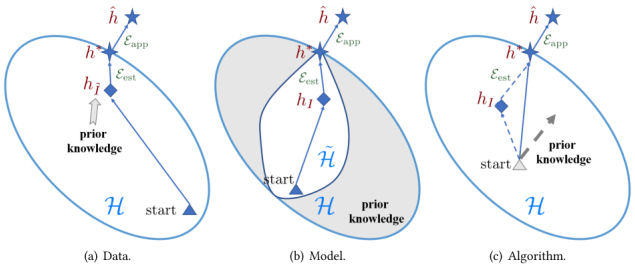


Image credit: Chen et al A Closer Look at Few-shot Classification ICLR 2019

Addressing ZSL/FSL



(a) Data. (b) Model. (c) Algorithm.

Data

- Learn to augment training data

Model

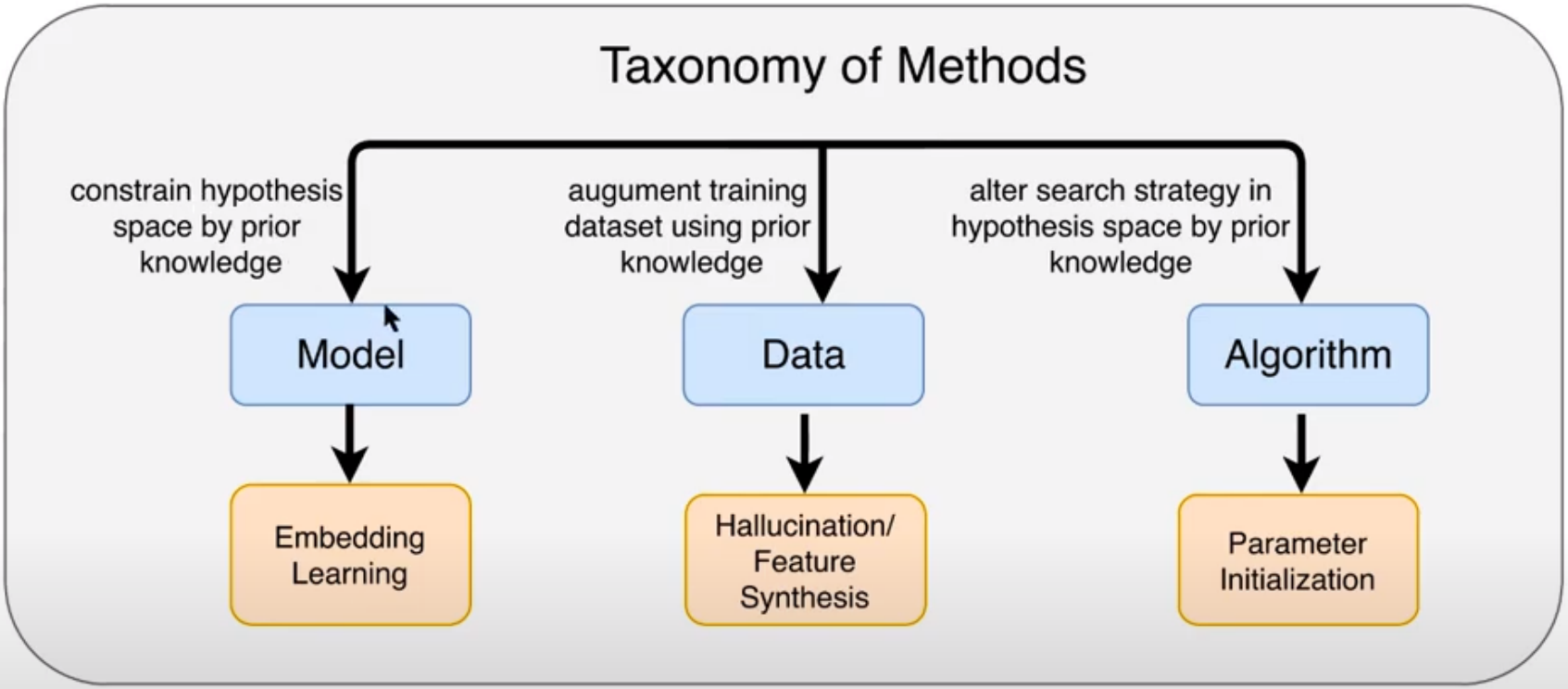
- Constrain complexity of H

Algorithm

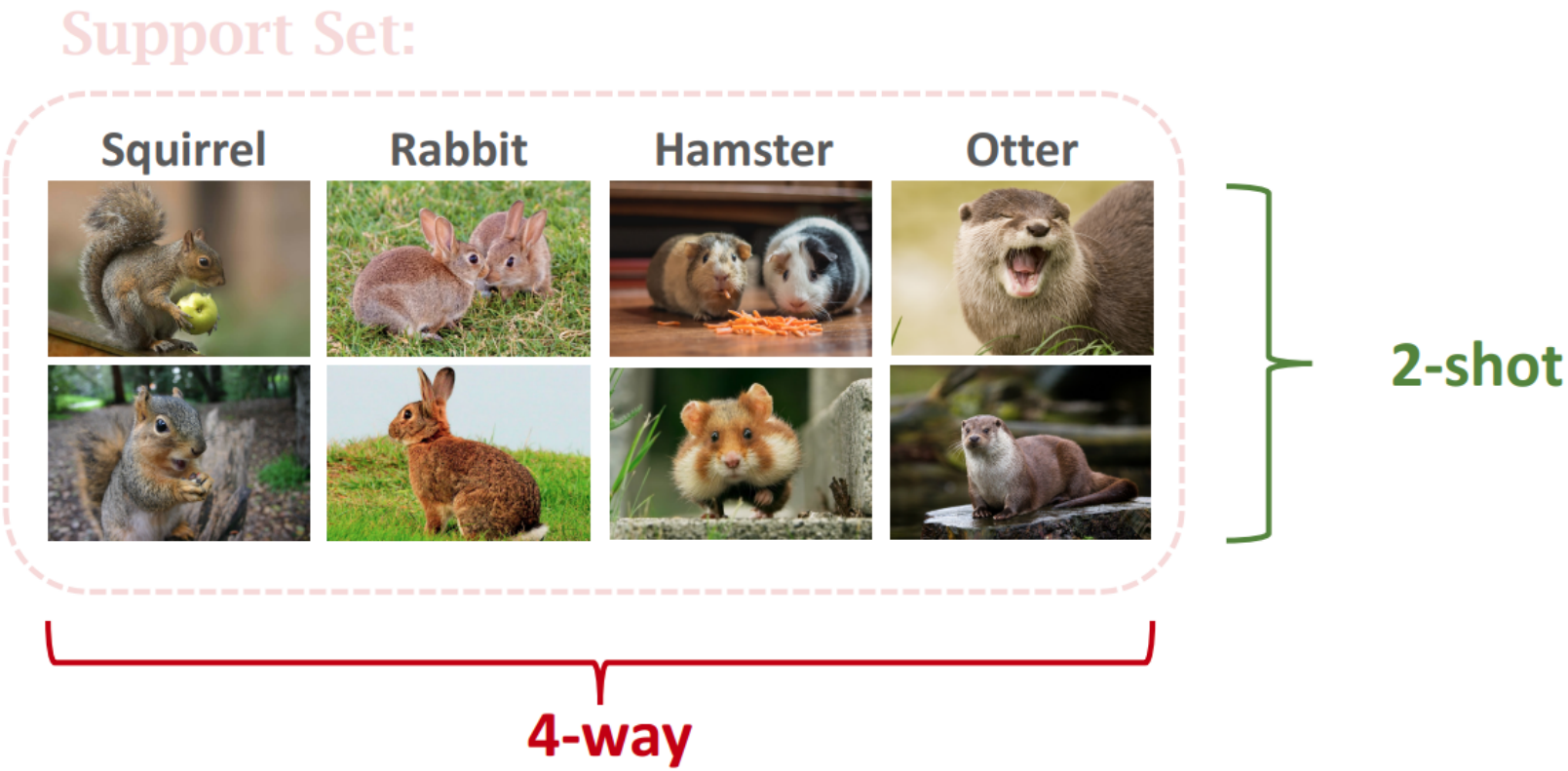
- Search for parameters θ for best hypothesis $h^\star$ in H using prior knowledge.

Image credit: WANG et al Generalizing from a Few Examples: A Survey on Few-shot Learning

Taxonomy of Methods



k-way n-shot Support Set



Basic idea

- First, learn a similarity function from large-scale training dataset.
- Then, apply the similarity function for prediction.
  - Compare the query with every sample in the support set.
  - Find the sample with the highest similarity score.

Basic idea

- First, learn a similarity function from large-scale training dataset.
- Then, apply the similarity function for prediction.
  - Compare the query with every sample in the support set.
  - Find the sample with the highest similarity score.

Support Set:

Basic idea

- First, learn a similarity function from large-scale training dataset.
- Then, apply the similarity function for prediction.
  - Compare the query with every sample in the support set.
  - Find the sample with the highest similarity score.

Basic idea

- First, learn a similarity function from large-scale training dataset.
- Then, apply the similarity function for prediction.
  - Compare the query with every sample in the support set.
  - Find the sample with the highest similarity score.

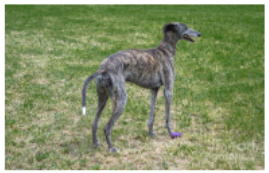
Support Set:

Basic idea

What is in the image?

Query:

sim = 0.2	sim = 0.1	sim = 0.03	sim = 0.05	sim = 0.7	sim = 0.5
Greyhound	Bulldog	Armadillo	Pangolin	Otter	Beaver



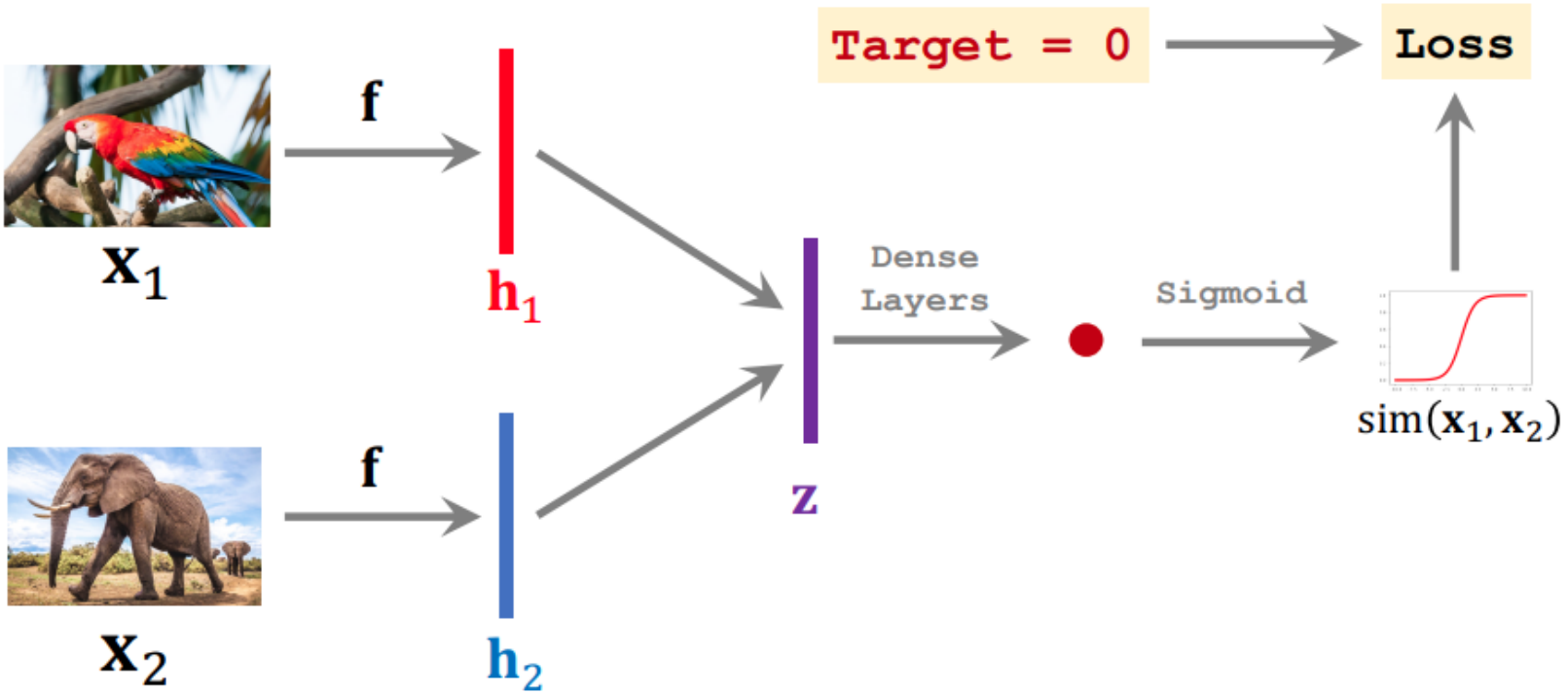



Training Data

Positive Samples

Negative Samples

Training Siamese Network



One-Shot Prediction

One-Shot Prediction

Query:

Support Set:

Fox

Squirrel

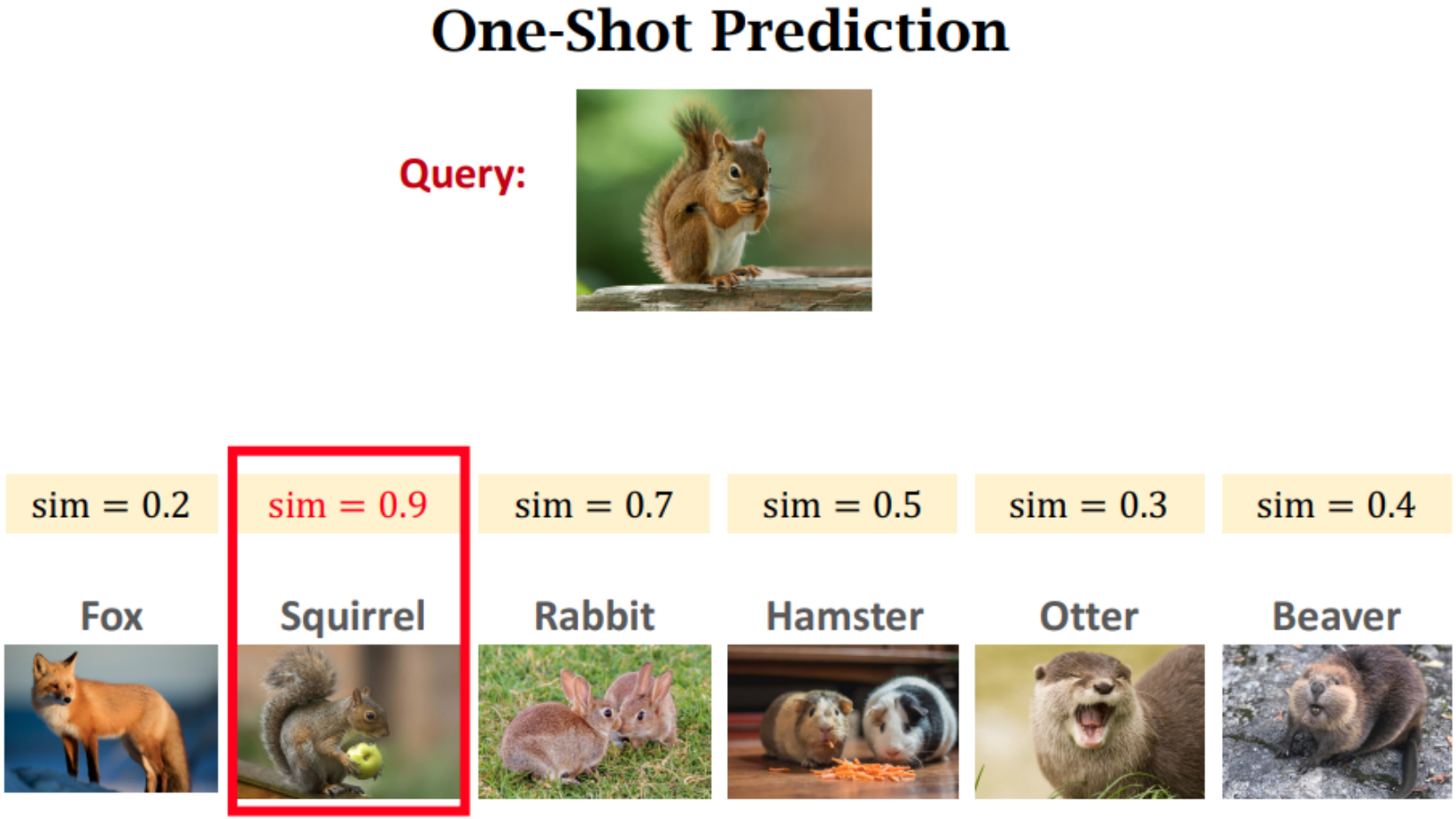
Rabbit

Hamster

Otter

Beaver

One-Shot Prediction



Data for Training Siamese Network

One-Shot Prediction

Query:

Support Set:

Fox

Squirrel

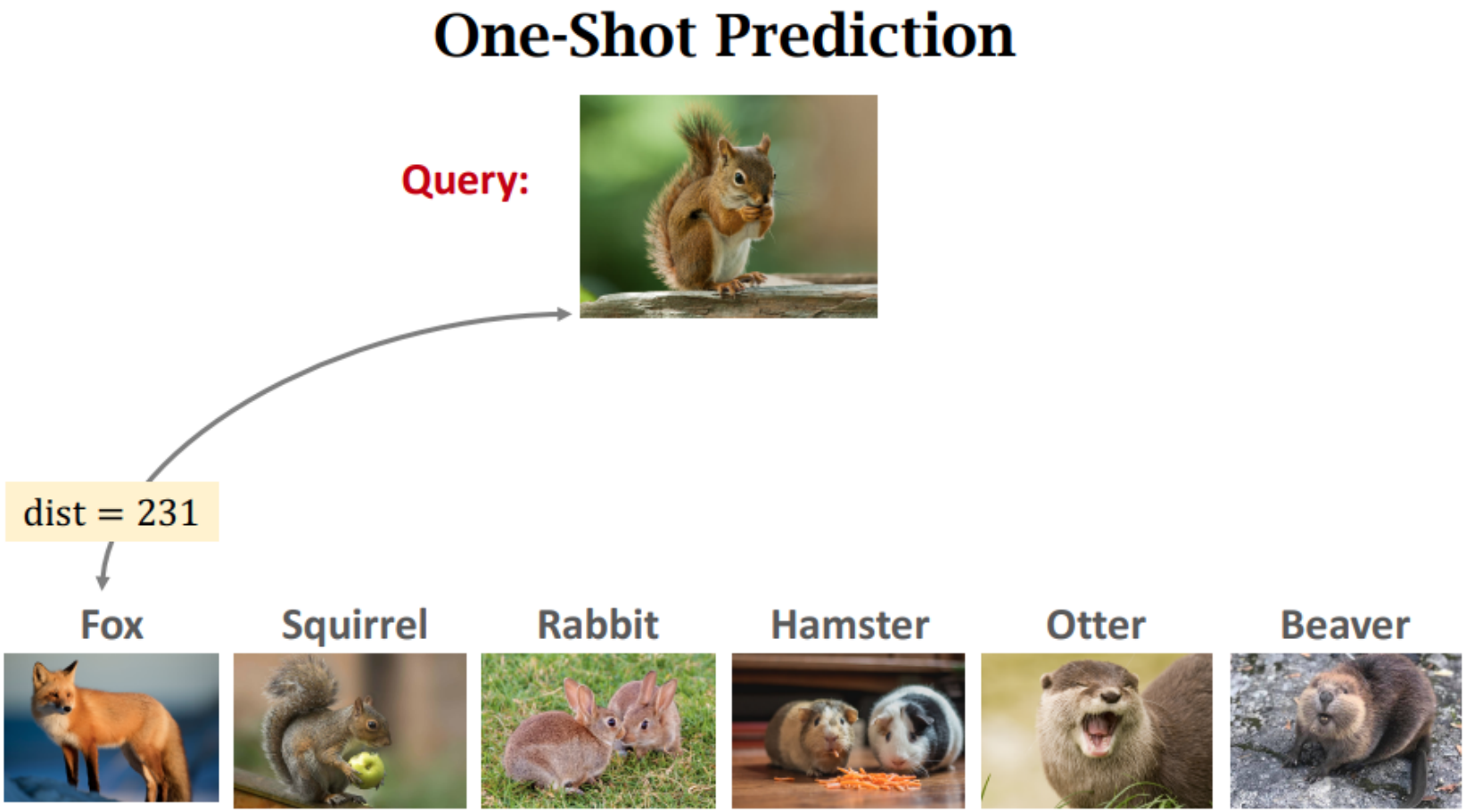
Rabbit

Hamster

Otter

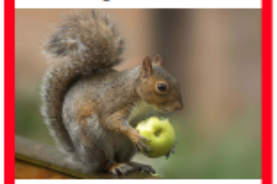
Beaver

One-Shot Prediction



One-Shot Prediction

Query:

dist = 231	dist = 19	dist = 138	dist = 76	dist = 122	dist = 94
Fox	Squirrel	Rabbit	Hamster	Otter	Beaver
					

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Basic Idea of Few-Shot Learning

- Train a Siamese network on large-scale training set.
- Given a support set of k-way n-shot.
  - k-way means k classes.
  - n-shot means every class has n samples.
  - The training set does not contain the k classes.
- Given a query, predict its class.
  - Use the Siamese network to compute similarity or distance

Motivation Zero Shot learning

Problem

- Can you say, which of the creatures, presented in these photos is an "Aye-aye"?
- I think you probably have never seen them. And because of this, you cannot give an answer.
- Just like typical ML model. Never seen - can't predict. (much better than random)

Motivation Zero Shot learning

Problem

- Can you say, which of the creatures, presented in these photos is an "Aye-aye"?
- I think you probably have never seen them. And because of this, you cannot give an answer.
- Just like typical ML model. Never seen - can't predict. (much better than random)

Motivation Zero Shot learning

Problem

- Can you say, which of the creatures, presented in these photos is an "Aye-aye"?
- I think you probably have never seen them. And because of this, you cannot give an answer.
- Just like typical ML model. Never seen - can't predict. (much better than random)

Motivation Zero Shot learning

Problem

- But what if i give you some knowledge about "Aye-aye"?
- Let's say, it lives on land. In this case, only 4 answers remain.
- It's relatively small. Then there are 2 options left.
- And it doesn't have a long neck Ta-daaaa, we've found it!
- What just happened? You gained knowledge about the animal and only then you were able to give a prediction

Motivation Zero Shot learning

Problem

- But what if i give you some knowledge about "Aye-aye"?
- Let's say, it lives on land. In this case, only 4 answers remain.
- It's relatively small. Then there are 2 options left.
- And it doesn't have a long neck Ta-daaaa, we've found it!
- What just happened? You gained knowledge about the animal and only then you were able to give a prediction

Motivation Zero Shot learning

Problem

- But what if i give you some knowledge about "Aye-aye"?
- Let's say, it lives on land. In this case, only 4 answers remain.
- It's relatively small. Then there are 2 options left.
- And it doesn't have a long neck Ta-daaaa, we've found it!
- What just happened? You gained knowledge about the animal and only then you were able to give a prediction

Motivation Zero Shot learning

Problem

- But what if i give you some knowledge about "Aye-aye"?
- Let's say, it lives on land. In this case, only 4 answers remain.
- It's relatively small. Then there are 2 options left.
- And it doesn't have a long neck Ta-daaaa, we've found it!
- What just happened? You gained knowledge about the animal and only then you were able to give a prediction

Motivation Zero Shot learning

Problem

- But what if i give you some knowledge about "Aye-aye"?
- Let's say, it lives on land. In this case, only 4 answers remain.
- It's relatively small. Then there are 2 options left.
- And it doesn't have a long neck Ta-daaaa, we've found it!
- What just happened? You gained knowledge about the animal and only then you were able to give a prediction

SO HOW IS IT WORKING IN TERMS OF CLASSIFICATION?

- **Given:**
  - We have pictures with class labels - it is our train set.
  - We have pictures without class labels - it is our zero-shot set.
- **Result:**
  - We want to find class labels for zero-shot dataset even if these classes did not occur in the train set.
- **But... How?**
  - Remember previous example. Imagine that you have never seen a single cat in your life.
  - And then someone gave you his detailed description. From that moment, if you see a cat, you will recognize him. We can implement this logic with image embeddings and semantic relationship between classes.

SO HOW IS IT WORKING IN TERMS OF CLASSIFICATION?

- **Given:**
  - We have pictures with class labels - it is our train set.
  - We have pictures without class labels - it is our zero-shot set.
- **Result:**
  - We want to find class labels for zero-shot dataset even if these classes did not occur in the train set.
- **But... How?**
  - Remember previous example. Imagine that you have never seen a single cat in your life.
  - And then someone gave you his detailed description. From that moment, if you see a cat, you will recognize him. We can implement this logic with image embeddings and semantic relationship between classes.

SO HOW IS IT WORKING IN TERMS OF CLASSIFICATION?

- **Given:**
  - We have pictures with class labels - it is our train set.
  - We have pictures without class labels - it is our zero-shot set.
- **Result:**
  - We want to find class labels for zero-shot dataset even if these classes did not occur in the train set.
- **But... How?**
  - Remember previous example. Imagine that you have never seen a single cat in your life.
  - And then someone gave you his detailed description. From that moment, if you see a cat, you will recognize him. We can implement this logic with image embeddings and semantic relationship between classes.

SO HOW IS IT WORKING IN TERMS OF CLASSIFICATION?

- **Given:**
  - We have pictures with class labels - it is our train set.
  - We have pictures without class labels - it is our zero-shot set.
- **Result:**
  - We want to find class labels for zero-shot dataset even if these classes did not occur in the train set.
- **But... How?**
  - Remember previous example. Imagine that you have never seen a single cat in your life.
  - And then someone gave you his detailed description. From that moment, if you see a cat, you will recognize him. We can implement this logic with image embeddings and semantic relationship between classes.

SO HOW IS IT WORKING IN TERMS OF CLASSIFICATION?

- **Given:**
  - We have pictures with class labels - it is our train set.
  - We have pictures without class labels - it is our zero-shot set.
- **Result:**
  - We want to find class labels for zero-shot dataset even if these classes did not occur in the train set.
- **But... How?**
  - Remember previous example. Imagine that you have never seen a single cat in your life.
  - And then someone gave you his detailed description. From that moment, if you see a cat, you will recognize him. We can implement this logic with image embeddings and semantic relationship between classes.

SO HOW IS IT WORKING IN TERMS OF CLASSIFICATION?

- **Given:**
  - We have pictures with class labels - it is our train set.
  - We have pictures without class labels - it is our zero-shot set.
- **Result:**
  - We want to find class labels for zero-shot dataset even if these classes did not occur in the train set.
- **But... How?**
  - Remember previous example. Imagine that you have never seen a single cat in your life.
  - And then someone gave you his detailed description. From that moment, if you see a cat, you will recognize him. We can implement this logic with image embeddings and semantic relationship between classes.

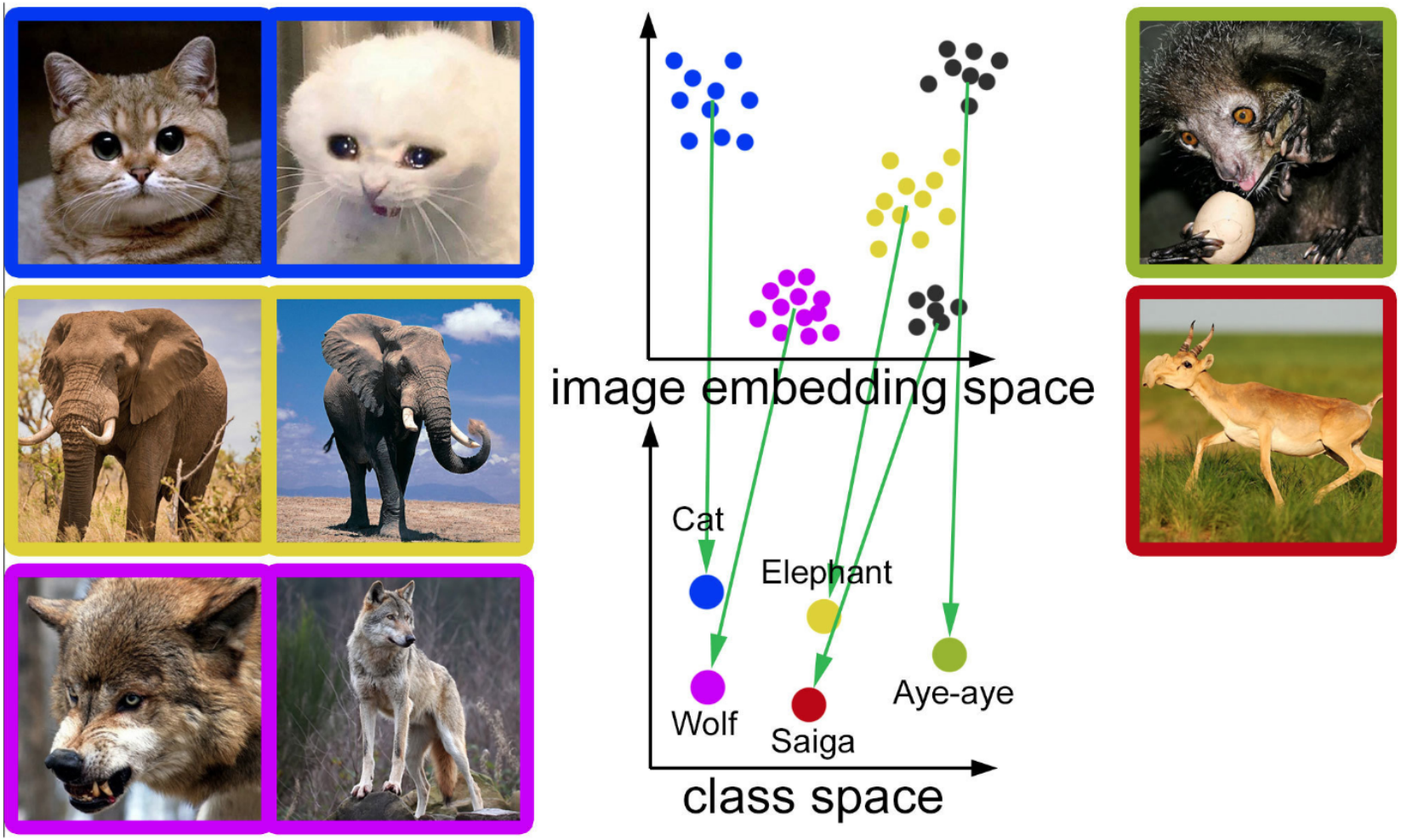
SO HOW IS IT WORKING IN TERMS OF CLASSIFICATION?

- **Given:**
  - We have pictures with class labels - it is our train set.
  - We have pictures without class labels - it is our zero-shot set.
- **Result:**
  - We want to find class labels for zero-shot dataset even if these classes did not occur in the train set.
- **But... How?**
  - Remember previous example. Imagine that you have never seen a single cat in your life.
  - And then someone gave you his detailed description. From that moment, if you see a cat, you will recognize him. We can implement this logic with image embeddings and semantic relationship between classes.

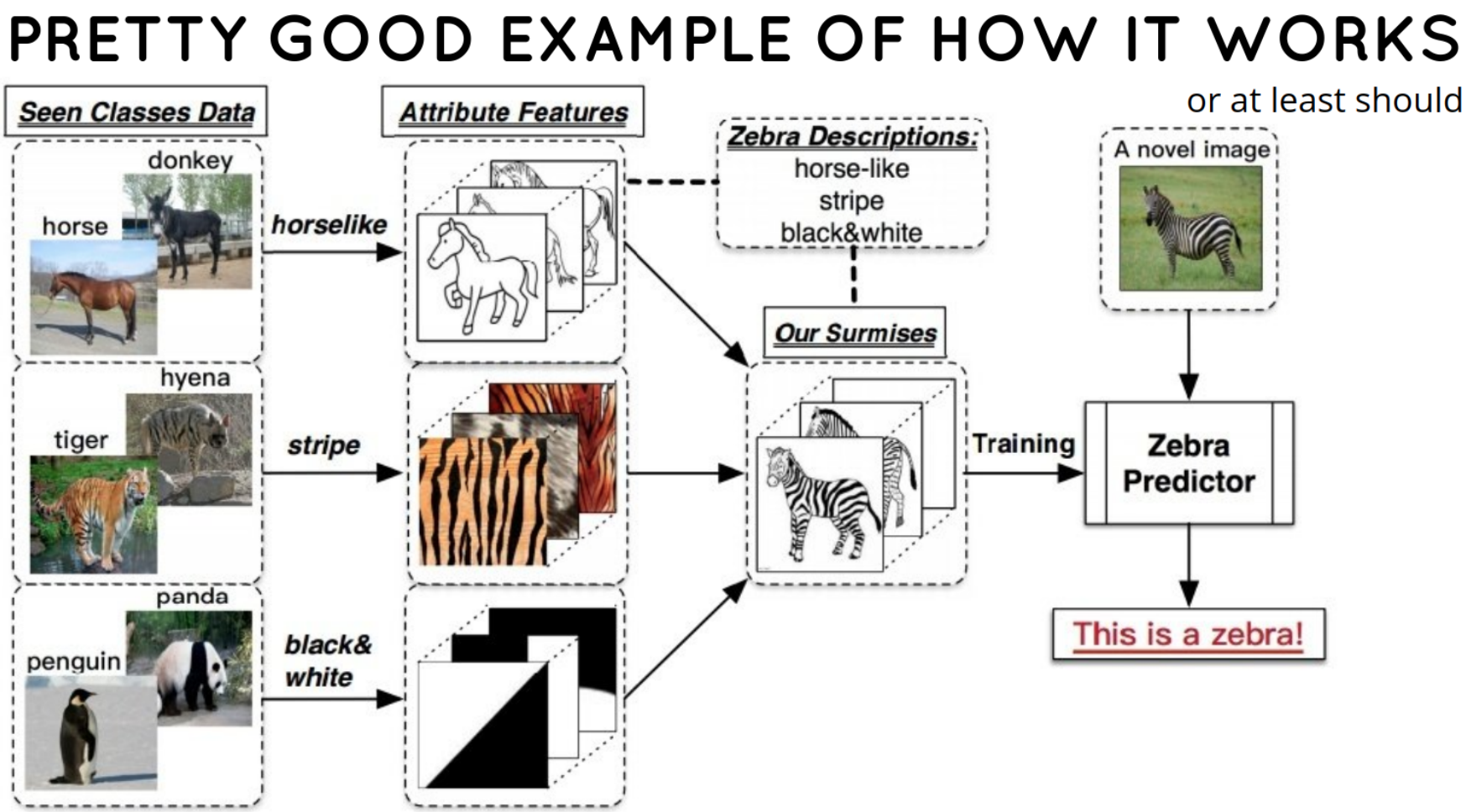
SO HOW IS IT WORKING IN TERMS OF CLASSIFICATION?

- **Given:**
  - We have pictures with class labels - it is our train set.
  - We have pictures without class labels - it is our zero-shot set.
- **Result:**
  - We want to find class labels for zero-shot dataset even if these classes did not occur in the train set.
- **But... How?**
  - Remember previous example. Imagine that you have never seen a single cat in your life.
  - And then someone gave you his detailed description. From that moment, if you see a cat, you will recognize him. We can implement this logic with image embeddings and semantic relationship between classes.

Motivation Zero Shot learning



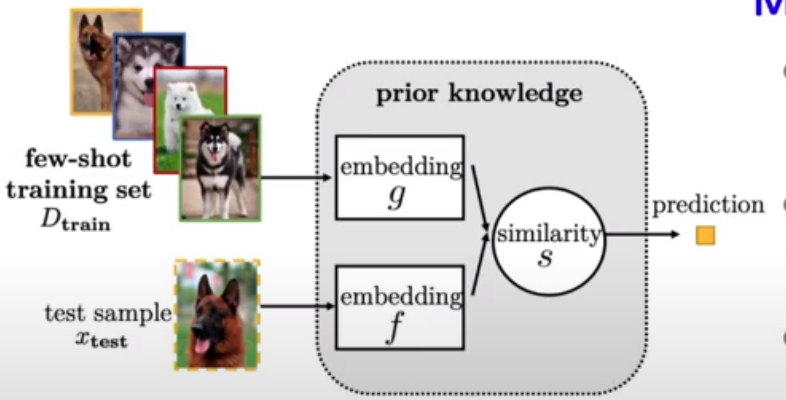
Zero Shot learning



Embedding learning method

Intuition:

- Address few-shot learning by "learning to compare"
- If model can determine similarity of two images (and perhaps corresponding semantics of classes), it can classify unseen input in relation to labeled instance seen during training

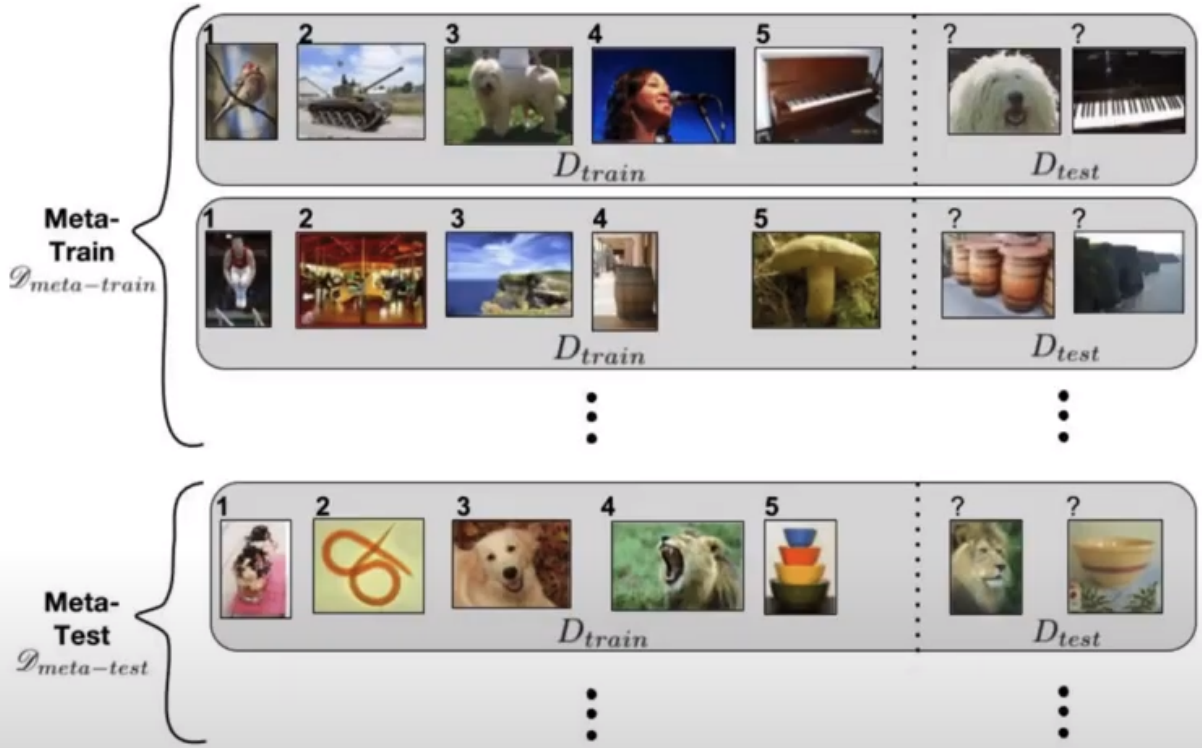


Method:

- Learn separate embedding functions for training samples D_{train} and test samples D_{test}
- Train sophisticated comparison models end-to-end via **meta-learning**
- At test time, predict based on comparing distance between x_{test} feature and training set features from each class

Problem setup: Meta learning

- **N-way-K-shot:** N different classes in D_{train} with K samples per class
- $D_{meta-train} = (D_{train}, D_{test})$ set $\rightarrow$ one task/episode
- Ensure $D_{meta-train}$ and $D_{meta-test}$ have disjoint/different classes



Problem setup: Meta learning

Learning Algorithm A

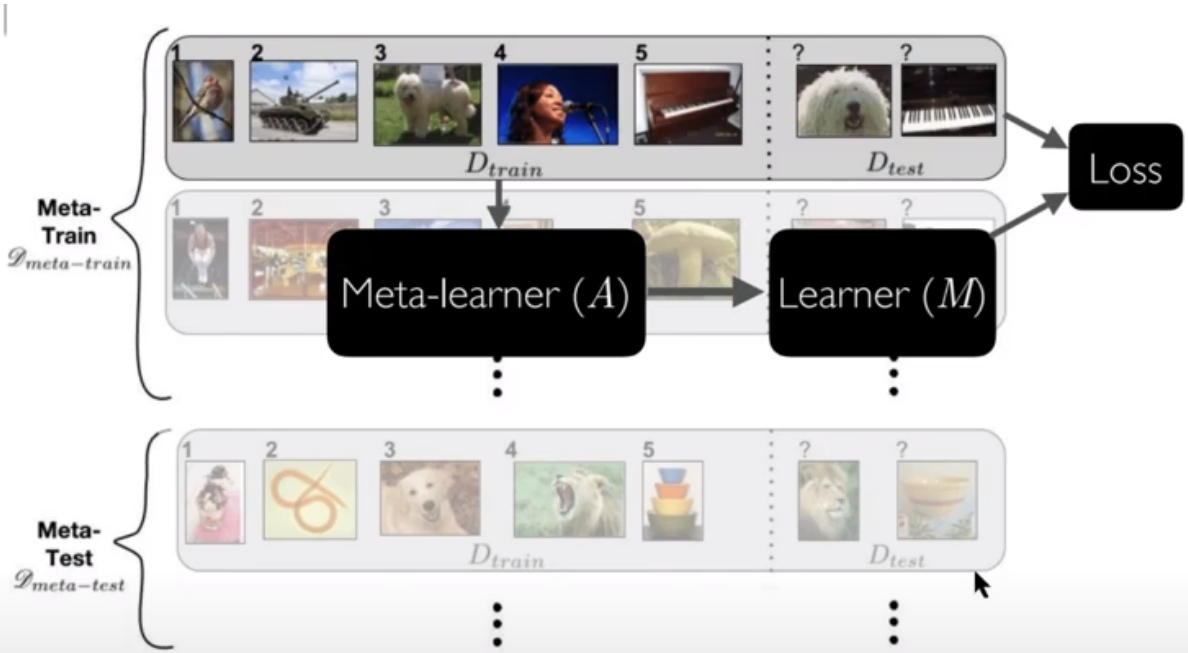
- **Input**: Training set $D_{train} = (x_i, y_i)_{i=1}^{I}$
- **Output**: Parameter θ model M (the **learner**)
- **Objective**: Good performance on $D_{test} = (x'_i, y'_i)$

Meta-Learning Algorithm

- **Input**: Meta-training set $D_{meta-train} = (D_{train}^{(n)}, D_{test}^{(n)})_{n=1}^{N}$ of tasks/episodes
- **Output**: Parameter Θ algorithm A(the **meta-learner**)
- **Objective**: Good performance on meta-test set $D_{meta-test} = (D_{train}^{\prime(n)}, D_{train}^{\prime(n)})_{n=1}^{N}$

Training setup: Meta learning

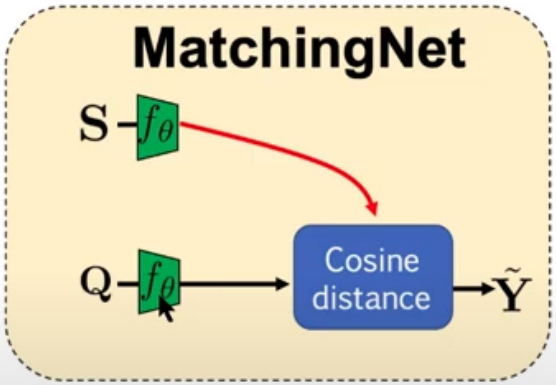
- **Training:** Repeat meta-loop for each task/mini-batch of tasks in $D_{meta-train}$ as shown
- **Inference:** On tasks/episodes in $D_{meta-test}$



Problem setup matches training and inference to enable generalization to new classes at test-time

Matching networks

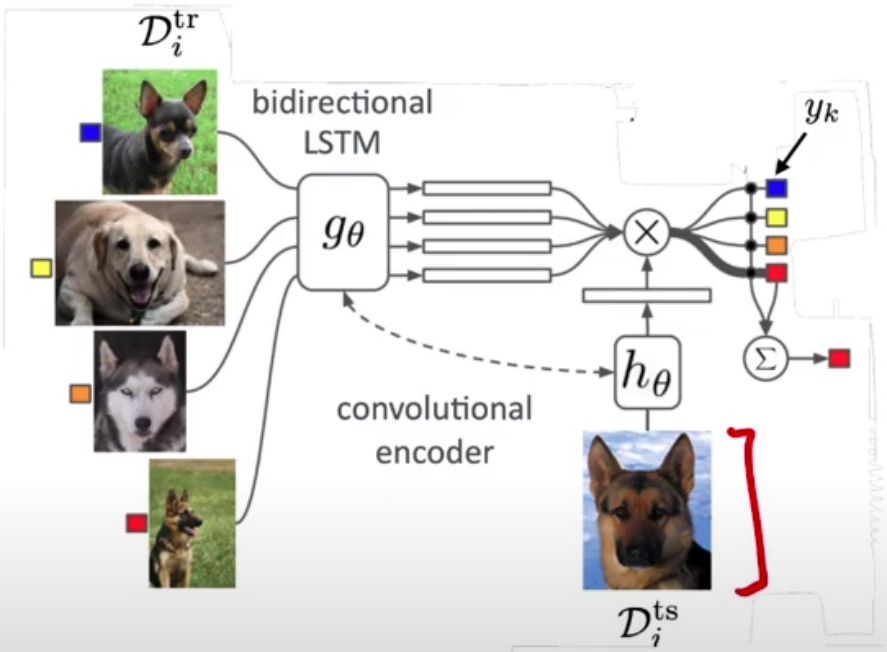
- **Parametric models:** Slowly learn model parameters from training examples; Require large datasets to avoid overfitting
- **Non-parametric models:** Allow novel examples to be rapidly assimilated; Robust to **catastrophic forgetting**



Idea: Combine best of both worlds

- **Training Phase:** Learn cosine similarity-based embedding models (parametric meta-learners)
- **Test Phase:** Use Nearest Neighbors (non-parametric) in learned embedding space

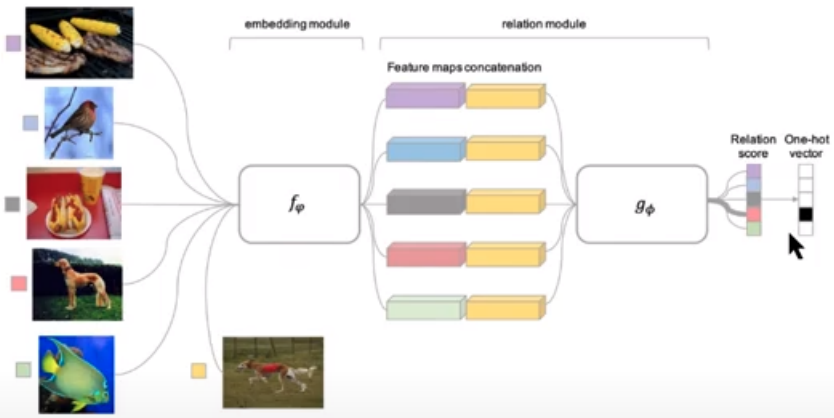
Matching networks



- $\hat{y}_{ts} = \sum_{i=1}^{k} a(\hat{x}, x_i) * y_i$
  where $a(\hat{x}, x_i)$ denotes the attention mechanism over examples
- Simplest form of attention mechanism $\implies$ softmax over cosine distances $c(.,.)$

$$a(\hat{x}, x_i) = \frac{e^{c(h(\hat{x}), g(x_i))}}{\sum_{j=1}^{k} e^{c(h(\hat{x}), g(x_j))}}$$

Relation network



One-shot Learning:

- Samples $x_j \in Q$ and $x_i \in S$ are fed into embedding module f_φ
- Feature maps $f_\varphi(x_j)$ and $f_\varphi(x_i)$ combined with concatenation operator $C(f_\varphi(x_j), f_\varphi(x_i))$ to extract relations
- Combined feature map fed into relation module g_ϕ, $\implies$ produces a scalar $\in (0, 1)$

$$r_{ij} = g_\phi(C(f_\varphi(x_j), f_\varphi(x_i)))$$

- **Objective:**

$$\varphi, \phi \leftarrow \arg\min_{\varphi,\phi} \sum_{i=1}^{m} \sum_{i=1}^{n} (r_{ij} - 1(y_i == y_j))^2$$

Relation network: Application to FS/ZSL

Few-shot Learning (K shot; $k > 1$)

- Use element-wise sum over embedding module outputs of samples from each training class (class feature map)
- Combine pooled class-level feature maps with query image feature map
  Number of relation scores for one query is always C (both one-shot or few-shot setting)

Zero-shot Learning:

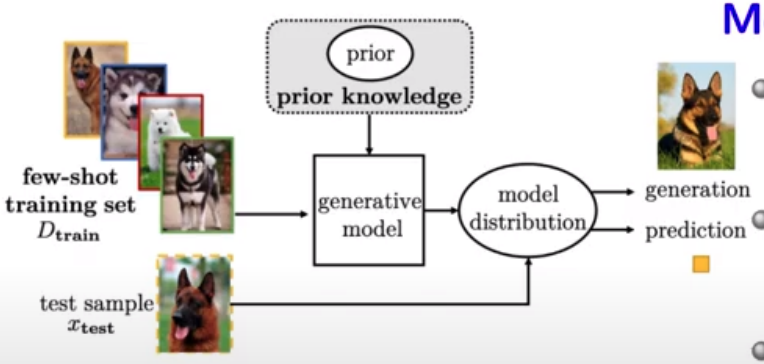
- Support set contains semantic class embeddings $\Longrightarrow$ vector v_c; instead of one-shot image for each class
- In addition to f_{φ_1} (for query images), introduce embedding module f_{φ_2} to handle semantic attributes

$$r_{ij} = g_\phi(C(f_{\varphi_2}(v_c), f_{\varphi_1}(x_j)))$$

Feature synthesis methods

Intuition:

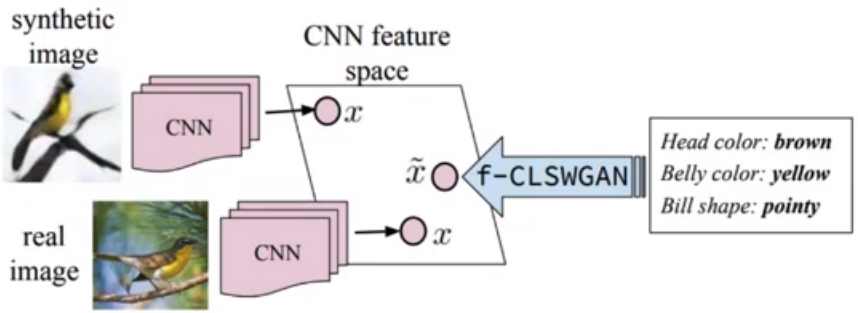
- Directly deal with data deficiency by "learning to augment"
- Learn a generative model to hallucinate new novel class data for data augmentation
- Reduce few/zero-shot problem to a standard supervised learning problem



Method:

- Learn a generator conditioned on meta-information using data in base classes
- Generate novel class features conditioned on unseen class meta-information
- Train a classifier on base class samples and generated novel class samples

Feature generating networks



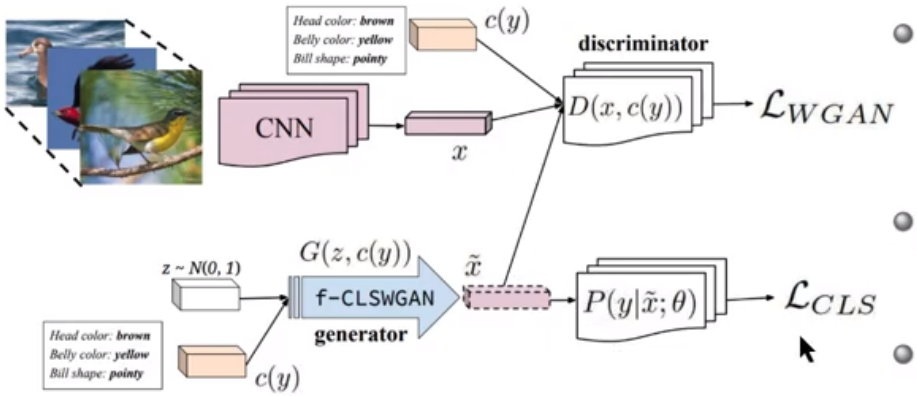
Method:

- Given train set S of seen classes, learn a conditional generator $G : Z \times C \rightarrow X$, which takes random Gaussian noise $z \in Z \subset R^{dz}$ and class embedding $c(y) \in C$, and outputs image feature $\tilde{x} \in X$
- To ensure $\tilde{x}$ are well-suited to train a discriminative classifier, minimize classification loss over generated features $\tilde{x}$

Extension to Few-shot Learning:

- For FSL, along with seen classes data set S, the training data also includes few labeled samples for each unseen class as well

Feature generating networks



Loss Formulation:

- **GAN Loss:**
$$\mathcal{L}_{WGAN} = E[D(x, c(y))] - E[D(\tilde{x}, c(y))] - \lambda E[(||\nabla_{\hat{x}} D(\hat{x}, c(y))||_2 - 1)^2]$$
- **Classification Loss:**
$$\mathcal{L}_{CLS} = -E_{\tilde{x} \sim p_{\tilde{x}}}[\log P(y|\tilde{x}; \theta)]$$
- **Final loss:**
$$L_{total} = \min_G \max_D \mathcal{L}_{WGAN} + \beta \mathcal{L}_{CLS}$$

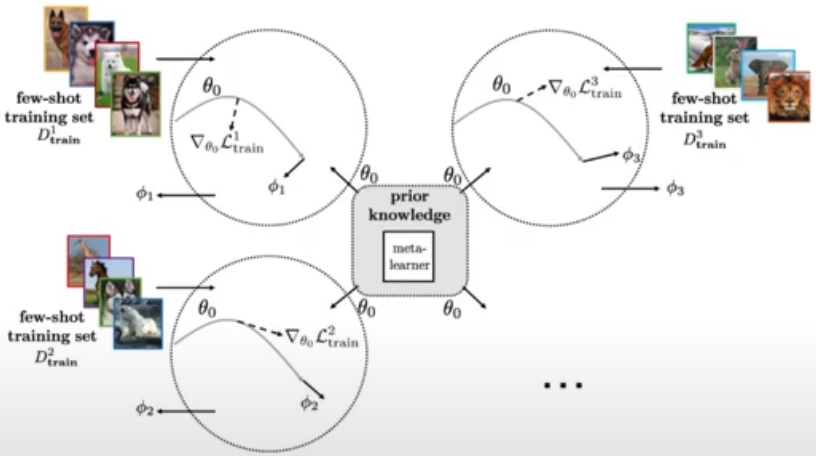
Training Classifier:

- Use pre-trained generator to generate samples novel/unseen class samples conditioned on class embeddings
- Train softmax classifier on train set and generated unseen class image features

Parameter Initialization methods

Intuition:

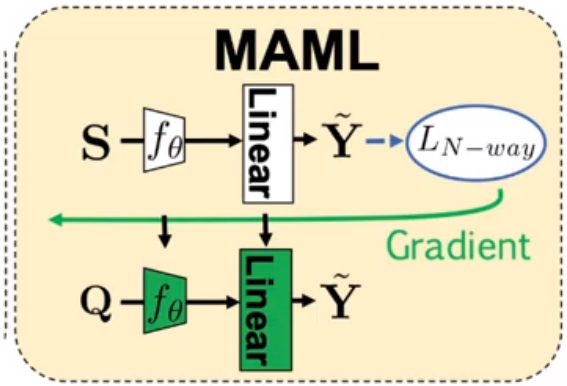
- Tackle the few-shot learning problem by "learning to fine-tune"
- Learn parameters that transfer via few-gradient steps (fine-tuning) to novel tasks



Method:

- For each task/episode $(D^i_{train}, D^i_{test})$, update task-specific parameters Φ_i to minimize $L(\theta, D^i_{train})$
- Update meta parameter θ to minimize $\sum L(\Phi_i, D^i_{test})$
- At test time, use few gradient steps to adapt classify novel classes

MAML



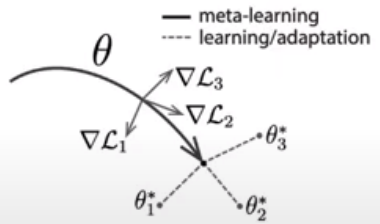


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

Key Idea: Acquire task-specific parameters (Φ_i) through optimization

Formulation

- Learn prior such that model can adapt to new tasks
  One form of prior knowledge: **Parameter Initialization**
- **Task-specific Update:**
  $\theta_i' = \theta - \alpha\nabla_\theta L(\theta, D^i_{train})$ (Single or multiple SGD updates)
- **Meta Fine-tuning:**
  $\theta = \theta - \beta\nabla_\theta \sum_i L(\theta_i', D^i_{test})$
  $\theta = \theta - \beta\nabla_\theta \sum_i L(\theta - \alpha\nabla_\theta L(\theta, D^i_{train}), D^i_{test})$
  (Second-order derivatives)

MAML:Algo

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
6: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2) or (3)
7: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
8: Sample datapoints $\mathcal{D}_i' = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the meta-update
9: **end for**
10: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3

Next Class